Data Science Self-Assessment

Galvanize Inc.

Contents

How to Use This Document

This document is designed to give you an idea of the baseline of Python and SQL knowledge required to apply for the Data Science Immersive program. If understanding any of the scripts included in

this PDF is challenging, we encourage you to take the time to study Python and/or SQL before beginning the application process. For a list of free Python and SQL resources, please refer to the DSI Study Resources PDF.

This document starts with some simple python statements which you should be able to evaluate without actually executing. We then proceed to more advanced challenges that will require a solid understanding of strings, lists, sets, dictionaries, file I/O, and functions. We then end the self assessment with a variety of SQL statements you should be comfortable with.

Spot the Differences

Without running the scripts, can you tell what the output will be? If you have some Python or programming background, this section should take very little time.

For Loops

```python
# Script 1
list_num = [1,2,3]
for num in list_num:
    total = 0
    total += num
    print total
```

```python
# Script 2
list_num = [1,2,3]
total = 0
for num in list_num:
    total += num
    print total
```

```python
# Script 3
list_num = [1,2,3]
total = 0
for num in list_num:
    total += num
print total
```

For Loops in Functions

```python
# Script 1
def my_function1(my_list):
    output = []
    for item in my_list:
        output.append(item)
        return item

print my_function1(['cat', 'bad', 'dad'])
```

—

```python
# Script 2
def my_function2(my_list):
    output = []
    for item in my_list:
        output.append(item)
        return output

print my_function2(['cat', 'bad', 'dad'])
```

—

```python
# Script 3
def my_function3(my_list):
    output = []
    for item in my_list:
        output.append(item)
    return item

print my_function3(['cat', 'bad', 'dad'])
```

—

```python
# Script 4
def my_function4(my_list):
    output = []
    for item in my_list:
        output.append(item)
    return output

print my_function4(['cat', 'bad', 'dad'])
print my_function4(['cat', 'bad', 'dad'])
```

—

```python
# Script 5
output = []
def my_function5(my_list):
    for item in my_list:
        output.append(item)
    return output

print my_function5(['cat', 'bad', 'dad'])
print my_function5(['cat', 'bad', 'dad'])
```

Make a function

Functions, blocks of reusable code, keep your code modular, well organized and easily maintainable. You should try to keep your code organized in functions. Take a look at each of the following snippets of code and organize them into functions.

1. We want a function that takes a list of numbers and returns that list where 10 was added to each number.

```
list_num = [1,2,3]
list_add_10 = []
for num in list_num:
    list_add_10.append(num + 10)
print list_add_10
```

---

2. We want a function that takes in a list of strings and returns the list with the length of the words.

```
list_words = ['great', 'job', 'so', 'far']
list_length_words = []
for word in list_words:
    list_length_words.append(len(word))
print list_length_words
```

More Advanced Python Challenges

Practice, practice, practice: we encourage you to work through these challenges.

Challenge 1

Write a function that looks at the number of times given letters appear in a document. The output should be in a dictionary.

```
def letter_counter(path_to_file, letters_to_count):
    ''' Returns the number of times specified letters appear in a file

    Parameters
    ----------
    path_to_file: str
        Relative or absolute path to file of interest
    letters_to_count: str
        String containing the letters to count in the text

    Returns
    --------
    letter_dict: dict
        - key: letter
        - value: the count of that letter in the file
    The counting is case insensitive
```

```
    Example
    --------
    ```file.txt
    This is the file of interest. Count my vowels!
    ```
    >>> letter_counter('file.txt', 'aeiou')
    {'i': 4, 'e':4, 'o':2, 'u':1}
    '''
    pass
```

Challenge 2

Write a function that removes one occurrence of a given item from a list. Do not use methods `.pop()` or `.remove()`! If the item is not present in the list, output should be 'The item is not in the list'.

```
def remove_item(list_items, item_to_remove):
    ''' Remove first occurrence of item from list

    Parameters
    ----------_
    list_items: list
    item_to_remove: object
        The object to be removed form list_items

    Returns
    --------
    - if the item is in the list: list
        list with first occurrence of item removed
    - if the item is not in the list: str
        'The item is not in the list'

    Example
    --------
    >>>list_items = [1,3,7,8,0]
    >>>remove_item(list_items, 7)
    [1,3,8,0]
    '''
    pass
```

Challenge 3

The simple substitution cipher basically consists of substituting every plaintext character for a different ciphertext character. The following is an example of one possible cipher from http://practicalcryptography.com/ciphers/simple-substitution-cipher/:

- Plain alphabet : abcdefghijklmnopqrstuvwxyz

- cipher alphabet: phqgiumeaylnofdxjkrcvstzwb

```
def cipher(text, cipher_alphabet, option='encipher'):
    ''' Run text through a particular cipher alphabet

    Parameters
    ----------
    text: str
        Either the plain text to encipher, or the cipher text to decrypt
    cipher_alphabet: dict
        Dictionary specifying {'original_letter': 'cipher_letter'}
    option: str (default 'encipher')
        'encipher' (accept plain text and output cipher text)
        'decipher' (accept cipher text and output plain text)

    Returns
    --------
    cipher text by default,
    plain text if option is set to decipher

    >>> cipher('defend the east wall of the castle',
               'phqgiumeaylnofdxjkrcvstzwb')
    'giuifg cei iprc tpnn du cei qprcni'
    >>> cipher('giuifg cei iprc tpnn du cei qprcni',
                'phqgiumeaylnofdxjkrcvstzwb',
                option='decopher')
    'defend the east wall of the castle'
    '''
    pass
```

Challenge 4

Implement a function that counts the number of isograms in a list of strings.

- An isogram is a word that has no repeating letters, consecutive or non-consecutive.
- Assume the empty string is an isogram and that the function should be case insensitive.

```
def count_isograms(list_of_words):
    ''' Count the number of strings without repeating characters in a list

    Parameters
    ----------
    list_of_words: list of strings

    Returns
    -------
    count of isograms (as integer)

    >>>count_isograms(['conduct',  letter', 'contract', 'hours', 'interview'])
    1
```

```
    '''
    pass
```

Challenge 5

Write a function that returns a list of matching items. Items are defined by a tuple with a letter and a number and we consider item 1 to match item 2 if:

1. Both their letters are vowels (aeiou), or both are consonnants and,
2. The sum of their numbers is a multiple of 3

(1,2) contains the same information as (2,1), the output list should only contain one of them.

```
def matching_pairs(data_list):
    '''
    Parameters
    ----------
    data_list: as list of tuples (letter, number)

    Returns
    -------
    A list of the matching pair referenced by their index (index_A, index_B).
    Each pair should appear only once. (A,B) is the same as (B,A)

    >>> data = [('a', 4), ('b', 5), ('c', 1), ('d', 3), ('e', 2), ('f',6)]
    >>> matching_pairs(data)
    [(0,4), (1,2), (3,4)]
    '''
    pass
```

Getting Ready for the SQL Assessment

You should be able to write the SQL queries that use `SELECT`, `FROM`, `WHERE`, `CASE` clauses, aggregates, and `JOIN`s . To check your work, you can run your queries on w3school's site (http://bit.ly/1foSkgu)

Our Data

We will be querying the following tables.

Table 1: `flags`

name	country	w_prop	l_prop	adoption_date
"Tricolour"	"France"	2	3	1830
"Union Jack"	"United Kingdom"	1	2	1801
"The Star-Strangled Banner"	"USA"	10	19	1960
"Hinomaru"	"Japan"	2	3	1999

name	country	w_prop	l_prop	adoption_date
“NA”	“Brazil”	7	10	1992
“Jalur Gemilang”	“Malaysia”	1	2	1963

where `w_prop` is the width proportion and `l_prop` is the length proportion

Table 2: `countries`

country	capital	contient
“France”	“Paris”	“Europe”
“Malaysia”	“Kuala Lumpur”	“Asia”
“Brazil”	“Brasilia”	“South America”
“United Kingdom”	“London”	“Europe”
“Japan”	“Tokyo”	“Asia”
“USA”	“Washington DC”	“North America”
“Germany”	“Berlin”	“Europe”

Simple Queries on a Single Table

1. Use the `WHERE` clause to show the countries with a flag ratio of 2:3 (i.e. `w_prop` = 2 and `l_prop` = 3).
2. Use `IN` to check if an item is in a list and show the countries on a continent that is either Europe or North America.
3. Use `BETWEEN xxx AND xxx` to show names of flags and countries that have width proportion higher than 1 but lower than 8.
4. Use `LIKE 'X%'` to show countries that have an name that starts with ‘U’.
5. Use `CASE` to show countries, their capital and a column to indicate whether the continent is ‘Eurasia’ (i.e. Europe or Asia) or ‘Americas’ (North or South America). Add a filter to select countries with capitals that are at least 7 character long.

Build Queries with Aggregates

Aggregates include commands such as `DISTINCT`, `COUNT`, `SUM`, `GROUP BY`, `HAVING`, and `ORDER BY`. Try using these commands on the following questions!

1. Use `DISTINCT` to list the continents in the countries table - each continent should appear only once.
2. Use `COUNT` to see how many countries are in Europe.
3. Use `GROUP BY` to count how many countries are in each continent, with continents alphabetically ordered (hint: use `ORDER BY`).
4. Use `HAVING` to determine which continents are represented at least twice in the countries table.

Build Complex Queries on Multiple Tables

1. Use `JOIN` to display the capital, the country, and the flag name.
2. Use `JOIN` and `WHERE` to display the continents associated to the flags in the flags table when the flag has a name (i.e. not 'NA').
3. Use `JOIN` and `HAVING` to display continents that have at least 2 countries represented as well as the average adoption date of the flag (as `avg_date`).